SmileBoom Petit Developer

プログラミングマニュアル

プチデベロッパーによるプログラム開発

2013-06-10

Version 1.1.3

株式会社スマイルブーム発行

資料内の記載されている会社名、サービス名、製品名は一般に各社の登録商標または商標です。

目次

1 この資料について 5
1.1 本資料の対象者 5
1.2 ブラウザごとの対応状況 5
1.2.1 レベル 1 （プログラマーが特に意識しなくともアプリが意図したとおり動く） 5
1.2.2 レベル 2 （完全に動作させるにはプログラマーが意識してコードを書く必要がある） 5
1.2.3 レベル 3 （動作保証なし。積極的な対応もしない） 5

2 プチデベロッパーの使い方 6
2.1 新規作成（プロジェクトの設定ダイアログ） 6
2.2 Canvas への画像追加 7
2.2.1 画像素材を用意する 7
2.2.2 Canvas への画像の貼り付けと内部的な自動生成要素 7
2.2.3 Layer について 8
2.2.4 画像以外（矩形・文字・動画・ボタン）の追加 8
2.3 その他の機能 9

3 プチデベロッパーの構造とライブラリの関係 10
3.1 プチデベロッパーに内蔵されているライブラリ sbt.js の役割 10
3.2 生成されたアプリケーションの動作構造 10
3.2.1 全体をまとめて制御（Canvas スクリプトを利用） 10
3.2.2 部品ごとに制御（Gadget スクリプトを利用） 10

4 サンプルの改造 11
4.1 サンプルの選択とテストプレイ 11
4.2 プロジェクトのコピー 11
4.3 改造11

5 プログラムの考え方 12
5.1 Canvas と Gadget 12
5.2 イベントに関するメソッドについて 12

6 具体的なプログラムコードの例 13
6.1 Canvas 変更（画面遷移） 13
6.2 Gadget の検索と動的生成 13
6.2.1 配置済み Gadget を探す 13

6.2.2 配置済み Layer を探す 13
6.2.3 指定した Layer 上に動的に Gadget を生成 13
6.2.4 既存 Layer 上に動的に Gadget を生成 13
6.3 動的な文字列変更プログラム 14
6.3.1 Test リソースそのものを書き換える 14
6.3.2 Animation 内の文字列を変更 14
6.3.3 特定の Animation の子アニメの文字列を変更 14
6.4 Bgm や SE の再生プログラム 14
6.4.1 Canvas スクリプトから BGM と SE の再生 14
6.4.2 Gadget スクリプトから BGM と SE の再生 14

7 プリセット Gadget の利用方法 15
7.1 Sbt.ButtonGadget 15
7.1.1 Sbt.ButtonGadget の動作 15
7.1.2 AnimationGroup の作成 15
7.1.3 Canvas への配置 15
7.1.4 パラメータの設定 16
7.1.5 イベントの設定 16
7.1.6 Sbt.ButtonGadget のプロパティ 16
7.1.7 Sbt.ButtonGadget のメソッド 16
7.2 Sbt.MovieGadget 17
7.2.1 Sbt.MovieGadget の動作 17
7.2.2 Canvas への配置 17
7.2.3 パラメータの設定 17
7.2.4 イベントの設定 17
7.2.5 Sbt.MovieGadget のメソッド 18

8 テストプレイとデバッグ 20
8.1 ファイル名の自動変換について 20
8.2 良くある単純なミス（タイプミス） 20
8.3 デバッグ用の便利な命令 20
8.4 Chrome を使ったデバッグ（23.0.1271.97 m） 21
8.4.1 ログを表示させるには 21
8.4.2 ソースを使ったトレース実行 21
8.4.3 複数 TAB で起動している場合の注意事項 21
8.5 Firefox を使ったデバッグ（17.0.1） 22
8.5.1 ログを表示させるには 22
8.5.2 ソースを使ったトレース実行 22
8.6 Internet Explorer を使ったデバッグ（9.0.8112.16421） 23

8.6.1 ログを表示させるには 23
8.6.2 ソースを使ったトレース実行 23
8.6.3 localhost を使わずにテストした場合の注意事項 23

9 エクスポート 24
9.1 ローカルで実行時の注意事項 25

10 JavaScript オブジェクト 26
10.1 Math 26
10.1.1 プロパティ 26
10.1.2 メソッド 26
10.2 String 27
10.2.1 メソッド 27

11 Sbt クラス 28

1 この資料について

「プログラミングマニュアル – 基本編」（以降、本資料）は、プチデベロッパーを使ってアプリケーションを作成するプログラマー向けの資料となります。本製品では、リソース（画像や音などの素材）とプログラムをまとめて扱うことができるため、素材等の準備や複雑なアニメーション用のプログラムを自分で用意することなくアプリケーションを作成することに集中することができます。

1.1 本資料の対象者

本資料はプログラマーおよびプログラマーを目指す方を対象にし、プチデベロッパーを使って初歩的なプログラムを記述するための情報を記載しています。本資料を理解するためには、HTML5 および JavaScript 言語（または C 言語や Java 言語などの知識）を理解していることが望まれますが、本資料の説明を読み、実際にプログラムを書くことで必要最低限の JavaScript 言語の知識とコード記述ができるような内容を目指しています。本格的な JavaScript や HTML5 によるプログラムの記述については専用の書籍等を購入して学習されることをお勧めいたします。

＜望まれる知識＞

1. JavaScript の言語仕様と基本的な命令の使い方
2. HTML5 の Canvas に関するプログラム的な知識
3. ブラウザ上でアプリケーションを動作させるための WEB サーバーに関する知識
4. 画像や音などの素材ファイルに関する基礎知識

1.2 ブラウザごとの対応状況

HTML5 規格とは言え機種や OS によって実装状況は異なります。本製品を使ったアプリケーション開発の難易度を以下にまとめます。

1.2.1 レベル 1 （プログラマーが特に意識しなくともアプリが意図したとおり動く）

iPad（サウンド、ムービー周りはレベル 2 相当。iOS6 を想定）、iPhone（サウンド、ムービー周りはレベル 2 相当。iOS6 を想定）、Safari（最新版のみ想定。Mac）
Chrome（最新版のみ想定。Windows, Mac）、Firefox（最新版のみ想定。Windows, Mac）、InternetExplorer10

1.2.2 レベル 2 （完全に動作させるにはプログラマーが意識してコードを書く必要がある）

Chrome（Android 4.1 以上）、Internet Explorer9、Xbox360、WiiU、Windows Phone 7(doMouseMove が来ない)

1.2.3 レベル 3 （動作保証なし。積極的な対応もしない）

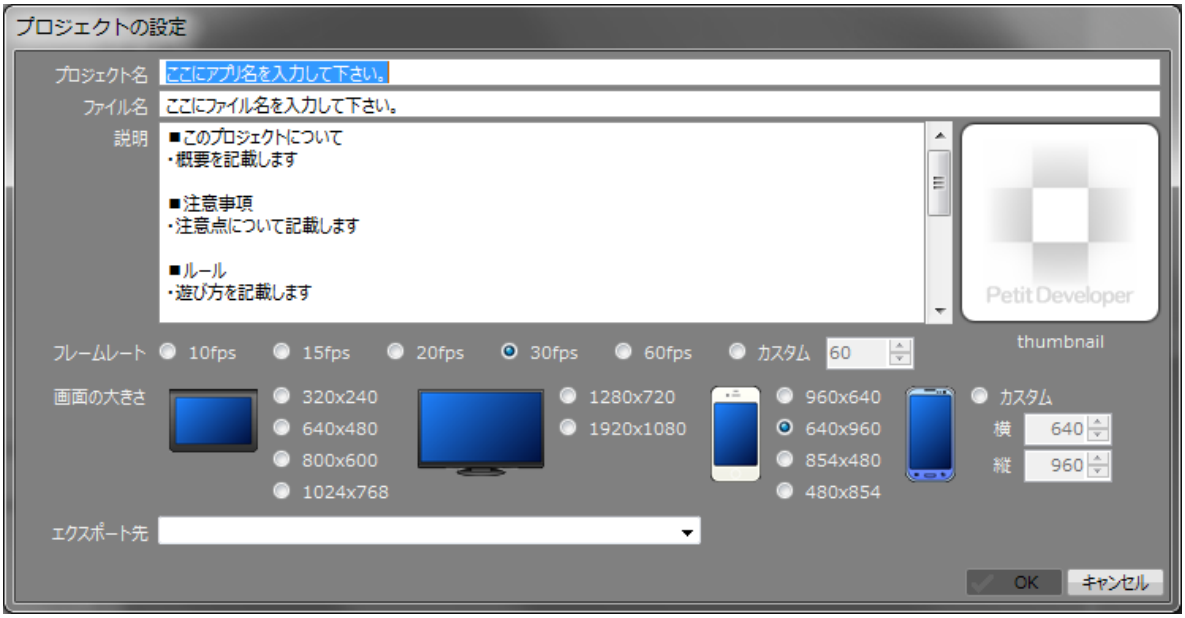
過去の iOS 搭載機（iOS5 以下）、Android 4.0 以下の標準ブラウザ、コンシューマー機のブラウザ（PS3, VITA, 3DS）、Opera、その他

2 プチデベロッパーの使い方

プログラマーがプチデベロッパーを使ってアプリケーションを作成する上で必要な最低限のツールの使い方を説明します。他の人との共同作業を行う場合にもツールの使い方を理解していれば意思疎通が早くなり作業効率が上がります。また、あらかじめプログラマーの方でデータの管理方法をツール上に登録しておきデザイナーやプランナーに中身を登録してもらうという指示方法も可能です。

2.1 新規作成（プロジェクトの設定ダイアログ）

ファイルメニューから「新規作成」を選ぶと表示されます。すべてを新規に作成したい場合はここから始まります。



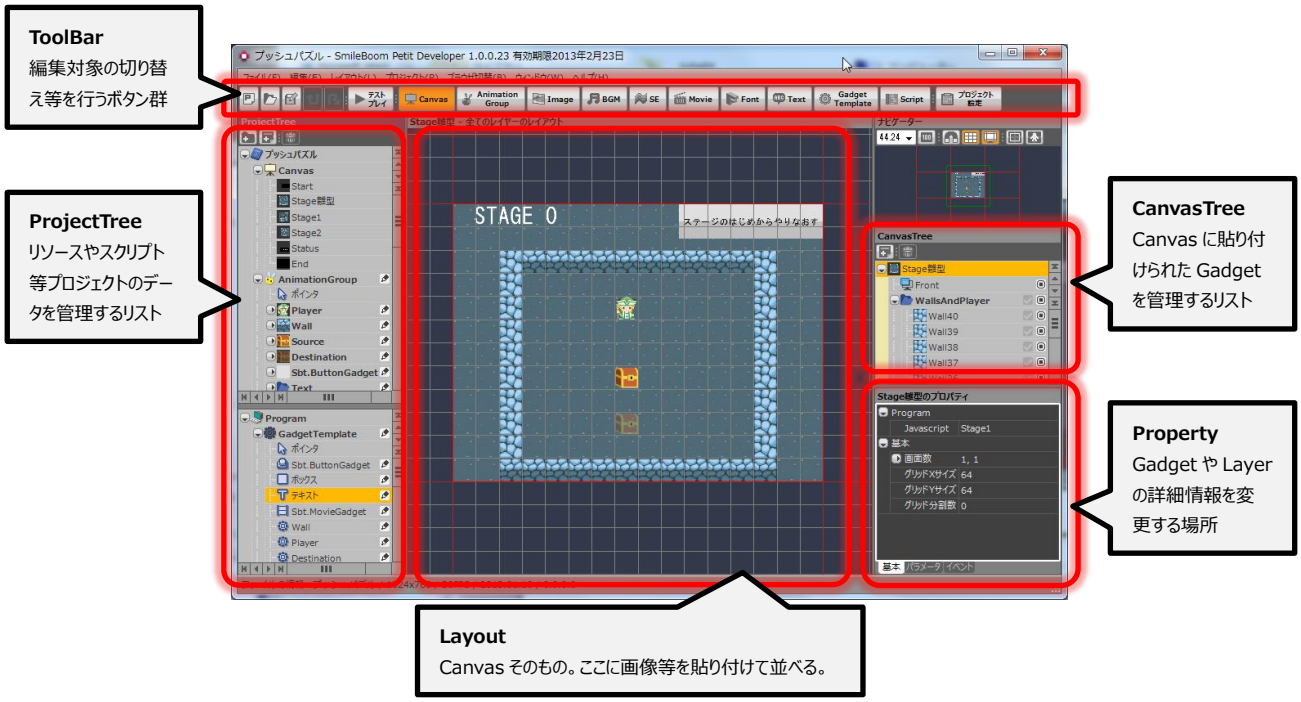
- **プロジェクト名** アプリケーションの名前を登録する場所
- **ファイル名** アプリケーションを保存するフォルダ名（ファイル名に日本語を嫌うサーバーを考慮すると英数字が理想）
- **説明** このアプリケーションについての説明（ファイルメニューの「開く」で情報が表示されます）
- **フレームレート** ブラウザ上での描画更新周期。数字が大きい方が滑らかに描画（実行環境の性能が低いと正しく更新されません）
- **画面の大きさ** 基本となる画面の解像度を選択（画素数が大きいほどデータ量も大きくなります）
- **エクスポート先** 完成したプロジェクトを外部サーバーへ転送するための転送先の情報を設定（エクスポート時に変更も可能）

すべての情報を登録し「OK」を押すと新しいプロジェクトのフォルダが作成されてプロジェクトに必要な最小限のファイルやデータが出力されます。

2.2 Canvas への画像追加

プチデベロッパーでアプリケーションを作る場合、表示の基本は Canvas となります。ブラウザ上には Canvas 経由で画像や文字を表示します。

ここでは、プチデベロッパーの Canvas を使って画面の表示レイアウトを作る方法を説明します。



2.2.1 画像素材を用意する

絵心があるプログラマーであれば、レタッチソフトやお絵かきツールなどを使って画像を用意し、音関係の扱いが得意であれば PC にマイクを接続して適当な音をサンプリングすることで SE 素材を準備します。絵や音を作ることが苦手な場合、プチデベロッパーのサンプルプロジェクト内に含まれるリソースから画像や音を流用します。また、ブラウザの検索エンジンを使って「フリー　画像　音　素材」などで検索すると様々なフリー素材を発見することができます。利用規約等を十分に読んで規約の範囲内でアプリケーション作成に活用してください。一般に個人の趣味で利用する場合は自由に利用できることが多いですが、販売を目的とした場合は何らかの許可や連絡が必要な素材が多いようです。各サイトの利用規約を熟読してご利用ください。なお、お客様がプチデベロッパーを作って配布したアプリケーションにおいて著作権侵害等の問題が発生してもスマイルブームは一切関知いたしません。

2.2.2 Canvas への画像の貼り付けと内部的な自動生成要素

エクスプローラー等から画像ファイルなどをドラッグし Canvas の Layout エリアにドロップすると現在選択されていた Layer に新しい Gadget が配置されます。デザイナー視点で考えた場合、貼り付けられた物の種類（画像・箱・テキスト・ムービー）によって貼られている物が異なる要素に見えますが、プログラマーから見た場合、Layer 上に貼りついているのはすべて Gadget と考えてください。Gadget は、Animation（描画要素）と Script（プログラム）を 1 つずつ管理するプチデベロッパーのコア要素です。プログラマーは表示されている物を意識せずにプログラムから表示物を扱うことができます。Canvas に画像を貼り付けると、Layer 上に Gadget が登録され、さらに内部的には AnimationGroup と Animation、および GadgetTemplatet が自動的に生成されます。自動生成された Animation を再び Canvas に貼りつけた場合は、Layer 上に Gadget が登録されるだけで他の要素は増えません。同じ画像を動かすプログラムは 1 つしか作らない仕様となっています。同じ画像でも別の処理で動かしたい場合は、Ctrl キーを押しながら貼り付けてください。エクスプローラーや Image リソースから同じ画像を Canvas に貼り付けた場合は、新しい Gadget と AnimationGroup と Animation、および GadgetTemplatet が生成されます。

2.2.3 Layer について

Canvas は、いくつかの Layer からなり、Layer ごとに登録された画像や文字を重ね合わせることで 1 つの画面を構成します。

2.2.3.1 FrontLayer

一番手前に表示される予約された Layer です。この Layer にユーザーが画像などを貼りつけることはできません。この Layer を選んで表示される Property には、フェードと SubCanvas という項目があります。フェードはこの Canvas が表示される時にフェードインで見え始める場合の色と時間を設定する要素で、SubCanvas は、例えばステージが複数あるゲーム等で Canvas ごとに持たせると扱いが面倒になるスコアや残り数等の共通表示要素を別の Canvas を重ねて表示することで実現している要素です。

2.2.3.2 BackLayer

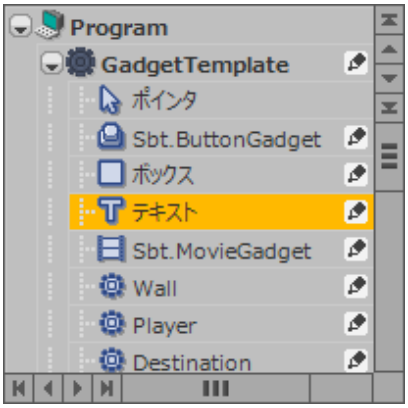
一番奥に表示される予約された Layer です。この Layer は背景用の画像 1 枚か単色を指定することができます。広大な画面の背景に常時表示される背景の模様がある場合は、この Layer に画像を指定すると楽です。

2.2.3.3 通常 Layer

任意の Gadget を貼り付けるための Layer です。通常、Canvas は複数の Layer を重ね合わせることで奥行き感や管理対象を分けて取り扱います。プログラムを書く場合も、対象となる Gadget がどの Layer に属しているかを意識して記述する必要があるため、新規作成時には Layer の扱いを管理しやすいように設計する必要があります。プッシュパズルのサンプルの CanvasTree を見ると Layer の分け方がわかりやすいと思います。なお、Layer 内にはフォルダを追加することもできます。フォルダはいくつかの Gadget をまとめて管理するグループ的な存在です。Layer とフォルダには座標の Property が用意されており、プログラムから値を変更することで Layer（フォルダ）単位でまとめて移動させることもできます。この機能をうまく利用すると多重スクロールのような表現が可能です。ジャンプアクションのサンプルで同様の処理を行っています。（ジャンプアクションは試用版には含まれていません）

2.2.4 画像以外（矩形・文字・動画・ボタン）の追加

画像以外の表示物は、Gadget にあらかじめ用意されています。



利用したい Gadget を選び、Canvas に貼り付けると Layer に Gadget が登録されて Canvas のプログラムから利用することができるようになります。配置した Gadget ごとに何か値を設定する必要がある場合は、CanvasTree から対象となる Gadget を選択し、表示された Property から数値等を変更してください。

2.3 その他の機能

本資料はプログラマー向けとして用意されているため、さらに詳しいツールの使い方は説明を省略します。Animation や Text などの他のリソースを作成または登録する方法については、ヘルプから「取扱説明書」を開いて確認してください。ヘルプから「API リファレンス」を開き、各クラスのメソッド名とツールのキャプション名を比較すると何をするための機能なのかを予測できると思います。

3 プチデベロッパーの構造とライブラリの関係

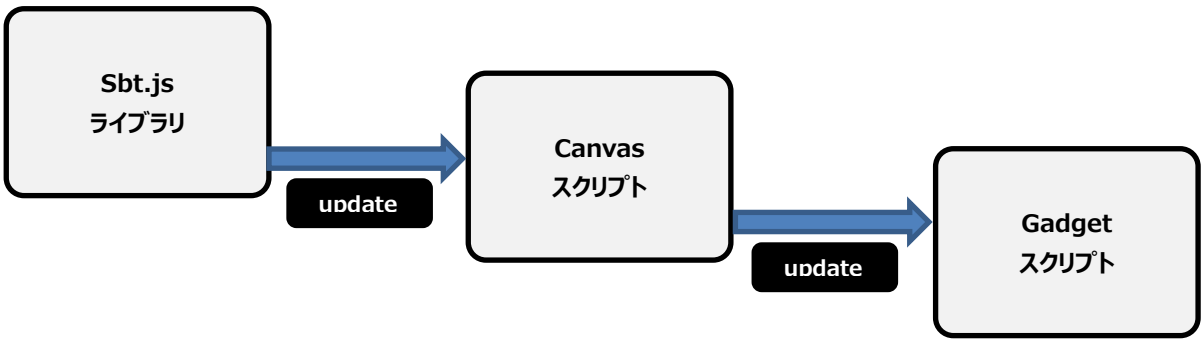
この章では、プチデベロッパーが管理する各種素材やスクリプト（プログラム）についての概念をプログラマー向けに説明します。プチデベロッパーに含まれるサンプルの画像や音を変更したり、いくつかのゲームで用意されているステージを追加して楽しむだけであれば、デザイナーやプランナー向けに用意された資料「取扱説明書」をご覧ください。

3.1 プチデベロッパーに内蔵されているライブラリ sbt.js の役割

テストプレイやエクスポートによって出力されるプログラムには自動的に sbt.js が結合されます。sbt.js は、プチデベロッパー上で作成した画面の配置やアニメーションをブラウザ上で効率よく表示するための描画エンジンやブラウザごとに仕様が異なる音の再生等を意識せずに利用できる機能などが組み込まれたライブラリです。ライブラリ内に用意されているクラスやメソッドなどの詳細な API 情報は別途用意されているドキュメントを参照してください。

3.2 生成されたアプリケーションの動作構造

プチデベロッパーで作成したアプリケーションは以下のような構造で実行されています。



3.2.1 全体をまとめて制御（Canvas スクリプトを利用）

ある Canvas が表示されている状況で、Canvas 上の Gadget をまとめて制御したい場合は、Canvas のスクリプトで個別の Gadget を制御するプログラムを記述します。この書き方は C や Java などの一般的なプログラムで main 処理を記述する感覚に近い方法です。

3.2.2 部品ごとに制御（Gadget スクリプトを利用）

プッシュパズルのサンプルのようにステージの雛型と配置する部品で構成されたアプリケーションを作る場合は、Gadget 側に処理を記述します。この方法は Gadget がプログラムを持ったスタンプのように扱えるため Canvas 側のプログラムよりも独立性が高く、ツール上の配置を変更することで新しいステージを追加するような作り方を実現しやすくなります。たとえば、シューティングの敵などは Gadget 側のスクリプトで処理を書いておけば、プログラムを知らない人がツールを使って Canvas に貼るだけでステージのレベルデザインを実現可能です。将来のバージョンでは、Gadget 単位の書き出しと取り込みをサポートし、プログラムと表示物がセットになった部品として扱えるようにします。

4 サンプルの改造

この章では、プチデベロッパーに含まれるサンプルを改造してプログラムの使い方についての基礎的な作業の流れを説明します。

4.1 サンプルの選択とテストプレイ

どのようなアプリケーションを作ることができるのかを確認するためにサンプルを実行することをお勧めします。サンプルを実行するためにはファイルメニューから「開く」を選び、サンプルの入ったフォルダをクリックして、サンプルのサムネイル画像が表示されたら試したいサムネイルをクリックしてツール上に読み込みます。読み込みが終わったら ProjectTree の上にある「テストプレイ」のボタンをクリックするか、ファンクションキーの 5 番（F5 キー）を押すと指定されたブラウザでアプリケーションが実行されます。ブラウザの指定が無い状態ではテストに使うブラウザを指定するためのダイアログが表示されるので好みのブラウザを選んでください。

4.2 プロジェクトのコピー

サンプルを直接変更してしまうと元々のプログラムが動かなくなる可能性もあるため、変更する前にプロジェクトをコピーすることをお勧めします。プロジェクトのコピーは、エクスプローラー等で対象となるプロジェクトをフォルダごとコピーして、フォルダ名とフォルダ直下にあるプロジェクトの管理ファイル（拡張子.sbtpx）を同じ名前に変更してください。

4.3 改造

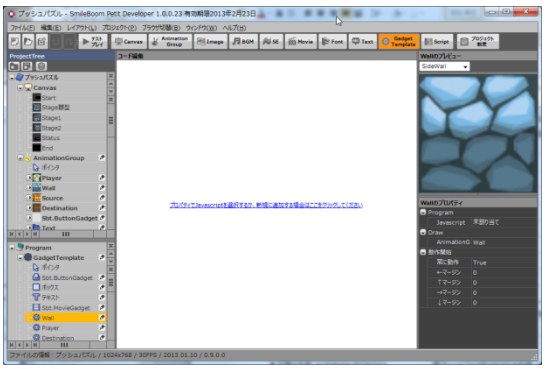
ProjectTree の下側にあるツリーから Script を選択して改造したいプログラムソースを選択します。ツール上にプログラムソースが表示され編集可能な状態になったことを確認し、日本語で書かれたコメントを見て数値などを変更してみましょう。移動量の変更や画面遷移を変更するところから始めると理解しやすいと思います。変更後 F5 キーで即座にブラウザ上で結果が確認できます。変更した後にブラウザ上で動かなくなった場合はテストプレイとデバッグの章を参考に問題を修正してください。

5 プログラムの考え方

この章ではプチデベロッパーを使ったプログラムの基本構造について説明します。アプリケーションを作成する上で共通化できる要素についてはライブラリ側であらかじめ処理が用意されているため、必要最低限のイベントに対応したプログラムを記述するだけでアプリケーションを作ることができます。

5.1 Canvas と Gadget

ツール上で Canvas を追加すると必ず Canvas 用のスクリプト（プログラム）が生成されます。もし、複数ある Canvas が個別の動きを必要としないのであれば、1 つの Canvas のスクリプトをすべての Canvas に割当てることもできます。Gadget のスクリプトについては、自動的に生成されません。Gadgetごとに制御したい場合は、対象となる Gadget を選択して「プロパティで JavaScript を選択するか、新規に作成する場合はここをクリックしてください」を押してスクリプトを追加してください。



5.2 イベントに関するメソッドについて

ツールによって生成されたスクリプトにはあらかじめ良く使われるメソッドがコメントアウトされた状態で登録されています。処理を組み込みたいイベントのコメントアウト部分を外してプログラムを追記してください。以下の要素が登録されています。

- 対象の名前.prototype. doInitialize　最初に呼び出される処理（初期化処理をここに記述します）
- 対象の名前.prototype.doUpdate　毎フレームごとに呼び出される処理（メイン処理をここに記述します）
- 対象の名前. prototype.doDraw　毎フレームごとに呼び出される描画の処理（通常あまり利用しません）
- 対象の名前.prototype.doMouseDown　タッチおよびマウスの左ボタンが押された時
- 対象の名前. prototype.doMouseMove　マウスカーソルが動いた時
- 対象の名前. prototype.doMouseUp　タッチおよびマウスの左ボタンが離された時
- 対象の名前. prototype.doDoubleClick　タッチおよびマウスの左ボタンがダブルクリックされた時
- 対象の名前. prototype.doTap　タップされた時
- 対象の名前. prototype.doFlick　フリックされた時

6 具体的なプログラムコードの例

良く使われる処理について具体的なコード例と共に説明します。なお、ここで説明されているコードや命令以外のライブラリのメソッドやプロパティについては、ヘルプから API リファレンスを呼び出して確認してください。

6.1 Canvas 変更（画面遷移）

ゲームアプリケーションでは、実行するとタイトル画面が表示されて何かボタンが押されるとステージが始まるような画面遷移が必要です。プチデベロッパーを使ったアプリケーションにおいても Canvas を切り替えることで画面遷移を実現します。

```
// タップされたら、キャンバスを切り替える
タイトル.prototype.doTap = function( location ) {
    // ステージ 1 に Canvas を切り替える
    this.app.changeCanvas("ステージ 1"); // this.app は Sbt.App のインスタンス
};
```

6.2 Gadget の検索と動的生成

ツール上で配置した Gadget は、必ず Canvas にぶら下がっています（正確には Canvas の Layer の下）配置済みの Gadget を利用する場合とプログラム内で動的に生成する場合のコードは以下のようになります。

6.2.1 配置済み Gadget を探す

```
var gadget=this.findGadget(“ガジェット名”);
```

6.2.2 配置済み Layer を探す

```
var layer=this.findLayer(“レイヤー名”);  //名称や Property が異なりますが Layer は Gadget と同じ扱いです
```

6.2.3 指定した Layer 上に動的に Gadget を生成

```
var gadget = this.createGadget( "レイヤー1", "enemyGadget" + this.enemyNo, [  x ,  y , 0 ], "ゴースト", "ゴースト", "移動" );
```

6.2.4 既存 Layer 上に動的に Gadget を生成

```
var ghostLayer = this.findLayer("ゴーストレイヤー");
var gadget = ghostLayer.createGadget( "enemyGadget" + this.enemyNo, [ x, y, 0 ], "ゴースト", "ゴースト", "移動" );
```

6.3 動的な文字列変更プログラム

プログラム内で文字列を変更する場合は以下のように記述します。

6.3.1 Test リソースそのものを書き換える

this.resource.setText("テキスト名", "テキスト");

リソース上のテキストは、新しい Gadget が生成される時の初期値として利用されます。すでに生成されている Gadget の Animation 内に対象となるテキストが含まれていたとしても、この命令を実行した結果テキスト内容は変化しません。

6.3.2 Animation 内の文字列を変更

this. findGadget(“スコア”).setText(score .toString());

Animation 内に複数の要素が含まれている場合、最初に発見された文字列を変更します。

6.3.3 特定の Animation の子アニメの文字列を変更

this.animation.findAnimation(“スコアアニメ名”).setText(score .toString());

複雑な階層を持つ Animation の中から特定の文字列を変更する場合に利用します。

6.4 Bgm や SE の再生プログラム

プログラム内で BGM や SE を変更する場合は以下のように記述します。

6.4.1 Canvas スクリプトから BGM と SE の再生

this.resource.playBgm("bgm");

this.resource.playSe("sd07");

6.4.2 Gadget スクリプトから BGM と SE の再生

Canvas 経由で再生させる必要があります。

this.canvas. resource.playBgm("bgm");

this.canvas.resource.playSe("sd07");

7 プリセット Gadget の利用方法

プチデベロッパーには標準機能として搭載されているプリセット Gadget があります。

7.1 Sbt.ButtonGadget

Sbt.ButtonGadget はシンプルなボタンの動作をする Gadget です。

7.1.1 Sbt.ButtonGadget の動作

ボタンを再現するための基本的な機能が含まれています。

1. **通常の AnimationGroup の他に、Animation の上に乗せる AnimationGroup の指定ができます**
   ※フレームを共通の AnimationGroup にし、個別に別の Animation を割り当てる等に使います
2. **タップした後に、専用イベントを呼び出します**
   ※タップした際の Animation と連動できるように、専用イベントを呼び出す時間を指定できます
3. **通常／タップされた時／無効状態の Animation を自動で切り替えます**
   ※上に乗せる AnimationGroup も同時に切り替えます
4. **無効状態では、タップを無効にします**

7.1.2 AnimationGroup の作成

Sbt.ButtonGadget 用の AniamtionGroup を作成する際には、プロジェクトの新規作成時に作られる Button をコピーするか、以下の三種類の Animation を持つ AniamtionGroup を作成して下さい。

1. **OFF**
2. **ON（ループのタイプに無限ループを指定しないで下さい）**
3. **DISABLE**

7.1.3 Canvas への配置

一般的な Gadget と同様に 2 通りの配置方法があります。

1. **AnimationGroup の Button を Canvas に配置**
2. **GadgetTemplate の Button を Canvas に配置**

必要であれば、AnimationGroup（OFF/ON/DISABLE）を変更

7.1.4 パラメータの設定

Sbt.ButtonGadget には、以下の専用パラメータがあります。

1. **topAnimationGroupId**

   Sbt.ButtonGadget の Animation の上に表示する AnimationGourp の名前指定

   ※指定する AnimationGourp には、OFF/ON/DISABLE の三種類の Animation が必要です

2. **clickFrame**

   Sbt.ButtonGadget のタッチ後、onClick が呼ばれるまでの待機時間（フレーム）

7.1.5 イベントの設定

Sbt.ButtonGadget には、以下の専用イベントがあります。

1. **onClick**

   配置されている Sbt.ButtonGadget がタップされて、clickFrame に指定されたフレーム数の後に呼ばれます。

   Sbt.ButtonGadget が enable フラグが false の時は、タップは無視されてこのイベントも呼ばれません。

7.1.6 Sbt.ButtonGadget のプロパティ

Sbt.ButtonGadget には、以下のプロパティがあります。

1. **enable**

   Sbt.ButtonGadget が有効か無効かのフラグ

7.1.7 Sbt.ButtonGadget のメソッド

Sbt.ButtonGadget には、以下のメソッドがあります。

7.1.7.1 setTopAnimationGroup(animationGroupId)

ButtonGadget の上に表示させる AnimationGroup の変更

animationGroupId：変更する AnimationGroup の名前

7.1.7.2 setEnable(enable)

ButtonGadget の有効／無効の変更

enable：有効にするかどうかのフラグ

7.2 Sbt.MovieGadget

Sbt.MovieGadget は Movie を再生するための Gadget です。

7.2.1 Sbt.MovieGadget の動作

動画を再生するための基本的な機能が含まれています。

1. **Movie をスクリプトで制御ができます（iPhone/iPod touch は再生開始のみ）**
2. **Movie の自動再生ができます（PC のみ）**
3. **Movie のフルスクリーン再生（ブラウザ全体での再生）ができます（PC のみ）**

7.2.2 Canvas への配置

一般的な Gadget と同様に 2 通りの配置方法があります。

1. **AnimationGroup の Movie を Canvas に配置**
2. **GadgetTemplate の Movie を Canvas に配置**

必要であれば、AnimationGroup を変更

7.2.3 パラメータの設定

Sbt.MovieGadget には、以下の専用パラメータがあります。

1. **movieId**

   再生する Movie の名前指定
2. **autoPlay**

   自動再生の指定（PC のみ）
3. **fullScreen**

   フルスクリーン再生の指定（PC のみ）

7.2.4 イベントの設定

Sbt.MovieGadget には、以下の専用イベントがあります。

1. **onEnded**

   Movie の再生が終わった時に呼ばれます。
2. **onError**

   Movie の再生に失敗した時に呼ばれます。
3. **onClick**

   配置されている Sbt.MovieGadget がタップされると呼ばれます。
4. **onClick**

   配置されている Sbt.MovieGadget がタップされると呼ばれます。

7.2.5 Sbt.MovieGadget のメソッド

Sbt.ButtonGadget には、以下のメソッドがあります。

7.2.5.1 setMovie(movieId, url)

Movie の変更（movieId で指定された Movie の URL が変更されます）

movieId：変更する Movie の名前

url：変更する Movie の URL（オプション）

7.2.5.2 setMovieUrl(url)

Movie の URL 指定での変更（現在設定されている Movie の URL が変更されます）

url：変更する Movie の URL

7.2.5.3 play()

Movie の再生

7.2.5.4 stop()

Movie の停止

7.2.5.5 pause()

Movie の一時停止

7.2.5.6 resume()

Movie の再生再開

7.2.5.7 seek(time)

Movie の再生位置の指定

time：再生させる位置(秒)

7.2.5.8 skip(time)

Movie の現在の再生位置からのスキップ

time：スキップさせる時間(秒)

7.2.5.9 isPlay()

Movie が再生中かどうかの取得

返り値：Movie の再生状態

7.2.5.10 isPause()

Movie がポーズ中かどうかの取得

返り値：Movie のポーズ状態

7.2.5.11 isStop()

Movie が停止中かどうかの取得

返り値：Movie の停止状態

7.2.5.12 getDuration()

Movie の長さの取得

返り値：Movie の長さ(秒)

7.2.5.13 getElapsed()

Movie の再生位置の取得

返り値：Movie の再生位置(秒)

8 テストプレイとデバッグ

この章では制作中のアプリケーションをブラウザで実行し問題点を発見するためのデバッグ方法について説明します。実行させるブラウザごとに操作方法が異なりますが、ここでは代表的な 3 つのブラウザについて説明します。

8.1 ファイル名の自動変換について

テストプレイやエクスポート時に、Canvas や Gadget 名に日本語が割り当てられていた場合、プチデベロッパーでは日本語によるファイル名を受け付けないサーバー用の対策としてファイル名を自動的に英語に変更してから出力します。ブラウザ上でのデバッグ時にソースファイルを表示した場合、ツール上の Canvas や Gadget の名称とは異なる名前になることをご了承ください。最初から英語で名前を付けておけば変換せずに利用します。

8.2 良くある単純なミス（タイプミス）

JavaScript は、インタープリタ型の言語のため実行するまで不具合が見えません。直前まで普通に動いていたプログラムが次にテストプレイすると動かなくなると言うことが良くあります。テストプレイでブラウザが起動した後に急に動かなくなった場合は、大体の場合文字列の入力ミスが原因のことが多いようです。残念ながらタイプミスを的確に発見するデバッグ方法はプログラムソースをじっくり見渡すしかありません。問題が発生している場所を特定するヒントとしては、各ブラウザのデバッグ機能（開発機能）を利用してログやエラーの状態を確認する方法があります。ブラウザごとのログの表示方法については以降の説明をご覧ください。

Javascript に限らず、**this.** を付け忘れること、変数名に i(アイ) や l(エル) を使っている部分を 1(イチ) と間違えることが多いようです。

8.3 デバッグ用の便利な命令

プログラム中の任意の位置に埋め込むことで、ブラウザ上で実行中にログ情報として出力された文字列を見ることができます。

Sbt.debugLog(msg)

汎用デバッグログ関数

例えば、数値変数 ox と oy の値を表示させたい場合は以下のようにコードを書きます。

（例） Sbt.debugLog(“offset(” + ox + “,” + oy + “)”);

また、プログラムの流れをとめて文字列を表示させる場合には alert を利用します。

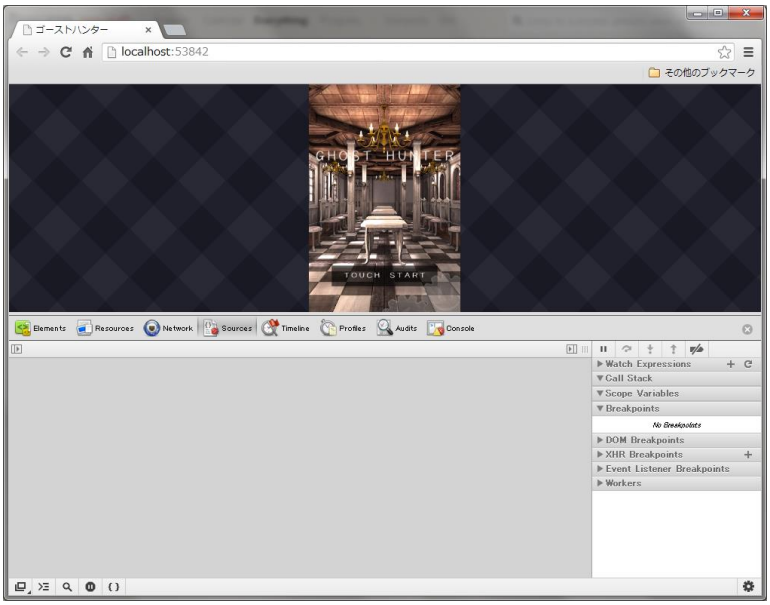
alert (msg)

警告ダイアログの表示

ブラウザ用のデバッグコンソールを使わずにブラウザ上にダイレクトにメッセージを表示させることができます。本当に処理が呼び出されているかどうかを確かめるためには、対象となる処理の直前に alert(“文字列”); を埋め込むことでブラウザ上にメッセージを表示させることができます。

8.4 Chrome を使ったデバッグ（23.0.1271.97 m）

Chrome では、F12 キーを押すと以下のように「デベロッパーツール」が表示されます。



8.4.1 ログを表示させるには

デベロッパーツールの左下にあるアイコンを押すとコンソールが表示されます。プログラム中で Sbt.debugLog()を使って表示した文字列はここに表示されます。タイプミス等で動かなくなった場合等にはブラウザ側からエラー情報が表示される場合があります。

8.4.2 ソースを使ったトレース実行

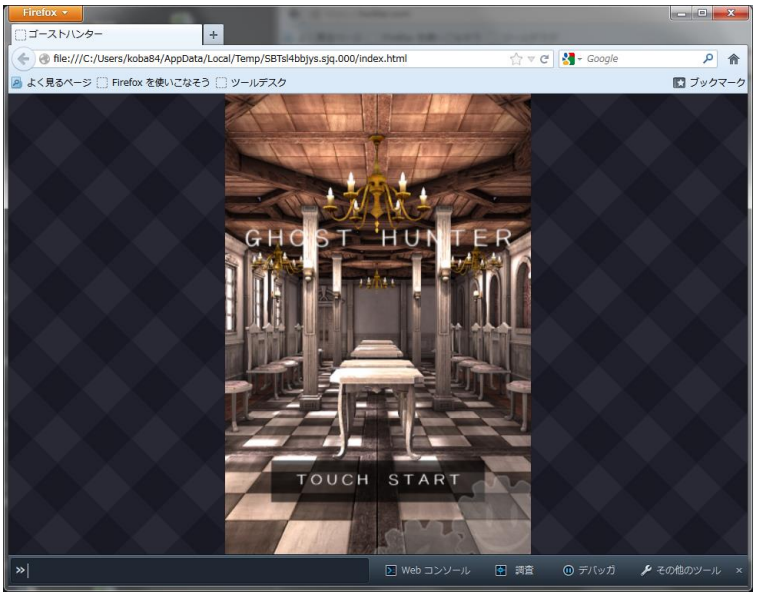
デベロッパーツールの機能としてプログラムソースを表示してブレークポイントを使ったデバッグ機能が搭載されています。この機能を使うと問題が発生しそうなプログラムにブレークポイントを仕込み実行し、ブレークポイントで止まったところからトレースするようなデバッグも可能ですが、この資料では詳しく説明いたしません。高度なデバッグを実現させるためには、「Chrome デベロッパーツール 解説」等で WEB 検索を行うか、関連する書籍等を参考に行ってください。

8.4.3 複数 TAB で起動している場合の注意事項

Chrome が起動し複数 TAB を開いている状態でテストプレイを実行するとエラーが出ることがあります。「ブラウザ切替」で localhost をチェックすると正しく起動します。

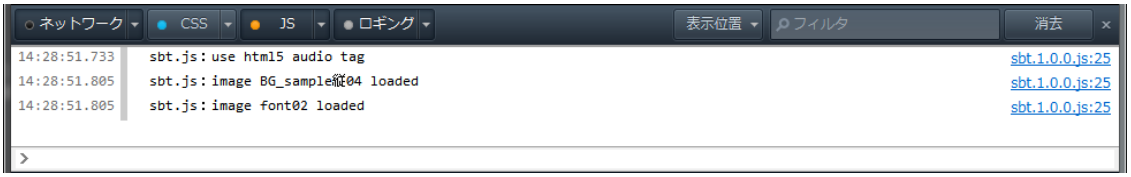
8.5 Firefox を使ったデバッグ（17.0.1）

Firefox では、Shift＋F2 キーを押すと「開発ツールバー」が表示されます。



8.5.1 ログを表示させるには

開発ツールバーにある「WEB コンソール」を押すとコンソールが表示されます。プログラム中で Sbt.debugLog()を使って表示した文字列はここに表示されます。

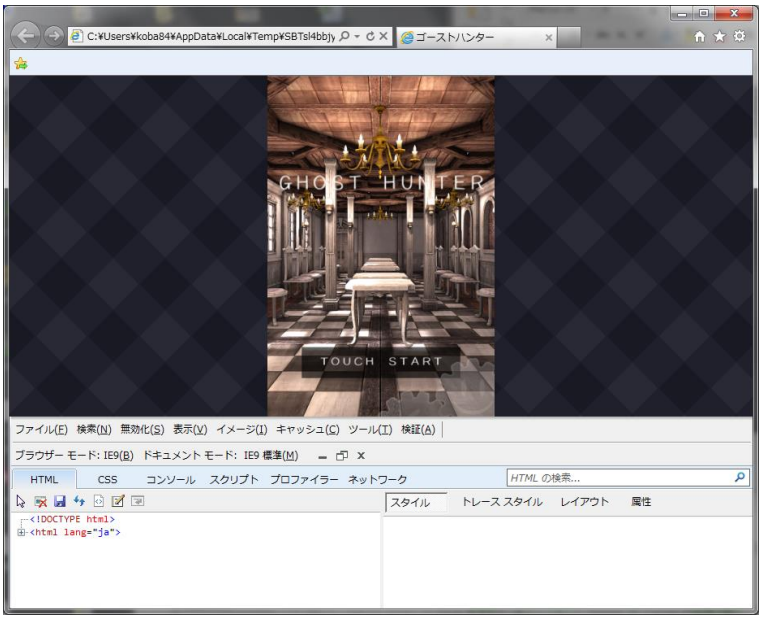


8.5.2 ソースを使ったトレース実行

開発ツールバーの「デバッグ」を押すとデバッガが表示されプログラムソースを表示してブレークポイントを使ったデバッグ機能が搭載されています。この機能を使うと問題が発生しそうなプログラムにブレークポイントを仕込み実行し、ブレークポイントで止まったところからトレースするようなデバッグも可能ですが、この資料では詳しく説明いたしません。高度なデバッグを実現させるためには、「Firefox 開発ツールバー デバッガ」等で WEB 検索を行うか、関連する書籍等を参考に行ってください。

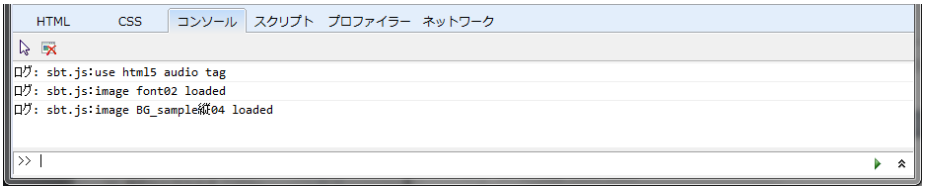
8.6 Internet Explorer を使ったデバッグ（9.0.8112.16421）

Internet Explorer では、F12 キーを押すことで「開発者ツール」が表示されます。



8.6.1 ログを表示させるには

開発者ツール上にある「コンソール」を押すとコンソールが表示されます。プログラム中で Sbt.debugLog()を使って表示した文字列はここに表示されます。

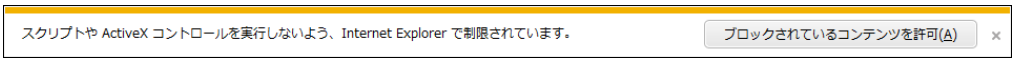


8.6.2 ソースを使ったトレース実行

開発者ツールの「スクリプト」を押すとプログラムソースを表示してブレークポイントを使ったデバッグ機能が搭載されています。この機能を使うと問題が発生しそうなプログラムにブレークポイントを仕込み実行し、ブレークポイントで止まったところからトレースするようなデバッグも可能ですが、この資料では詳しく説明いたしません。高度なデバッグを実現させるためには、「Internet Explorer 9 開発者ツール デバッグ」等で WEB 検索を行うか、関連する書籍等を参考に行ってください。

8.6.3 localhost を使わずにテストした場合の注意事項

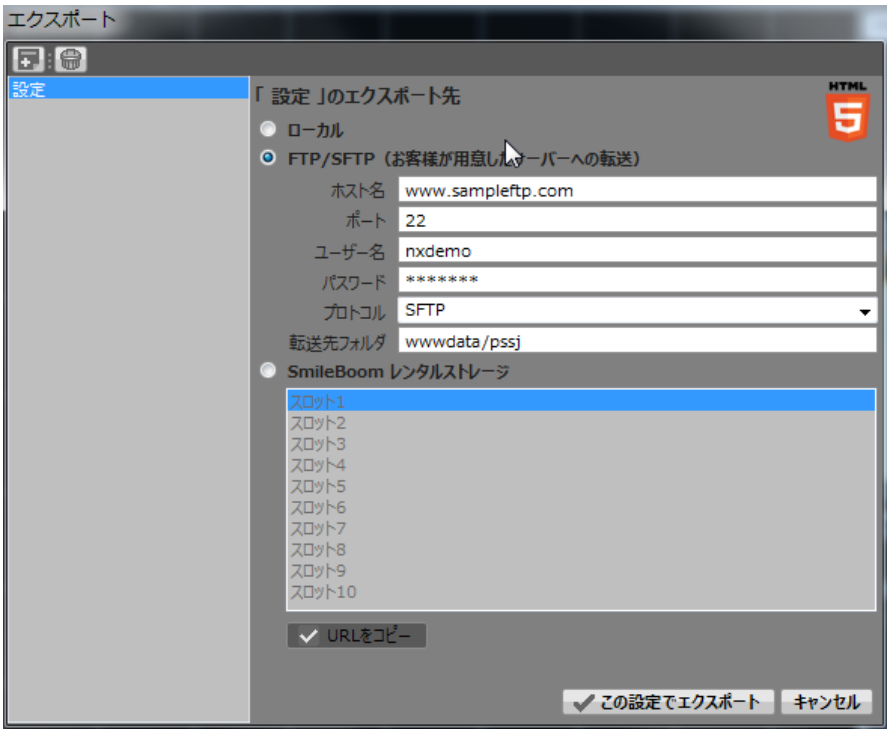
Internet Explorer を起動するとスクリプトと ActiveX に関する警告が表示されることがあります。「ブロックされているコンテンツを許可」を押して続行してください。



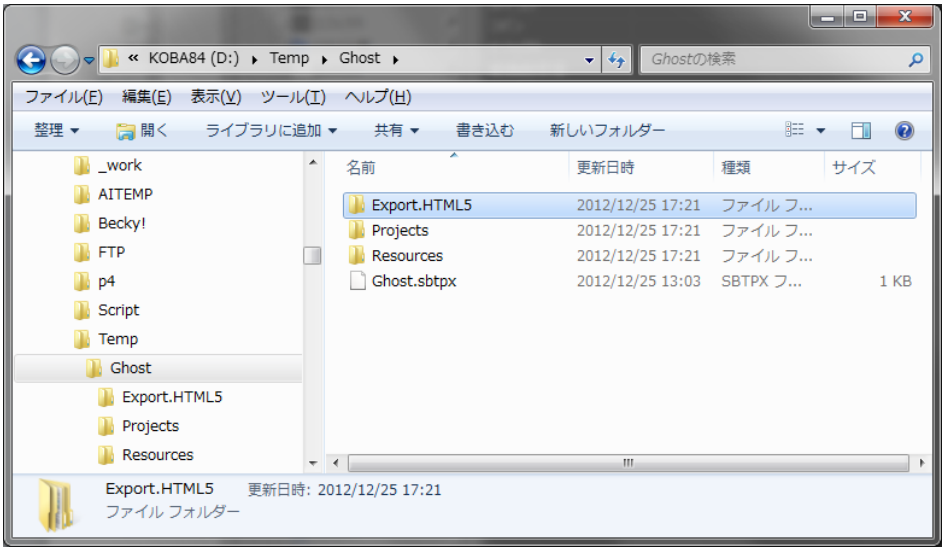
9 エクスポート

ローカル環境でブラウザを使ったテストプレイを繰り返し、完成したアプリケーションをサーバー上へ公開するときにエクスポート機能を利用します。

試用版では FTP/SFTP、スマイルブームストレージへのエクスポートは実行できません。



ローカル出力時は Project のフォルダ以下に Export.HTML5 というフォルダが生成されて出力されます。

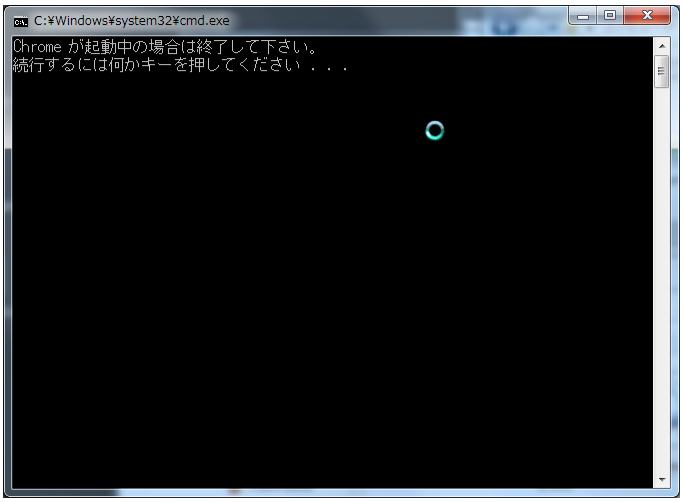


JavaScriptが実行可能なブログやホームページが利用できる場合、これ以下のフォルダ内容を転送することで作成したアプリケーションを利用することができるようになります。（製品版ではツールからダイレクトに転送可能です）

9.1 ローカルで実行時の注意事項

ブラウザによっては、ローカル上のファイルからは表示できない場合があります。プチデベロッパーが無い環境で完成したアプリを実行するためには、対応している主要なブラウザ向けのショートカット（Chrome、Firefox、IE）を使って起動させることができます。

これらのショートカットをダブルクリックすると自動的にブラウザが起動してアプリが動き始めます。なお、ブラウザの起動に問題がある場合は、警告メッセージを表示して終了します。指示に従ってブラウザを閉じる等の対応が必要です。



なお、出力されたフォルダ内に出力される以下のプログラムは、ブラウザを起動するためのランチャープログラムとなっています。

このプログラムを単独で利用することはありません。

BrowserLauncher.exe

10 JavaScript オブジェクト

利用頻度が高そうなオブジェクトとプロパティおよびメソッドを簡単にまとめたものです。詳しくは書籍や WEB 検索等でご確認ください。

10.1 Math

数値計算関連のオブジェクト。

10.1.1 プロパティ

プロパティ	説明
PI	円周率

10.1.2 メソッド

メソッド	引数	戻り値	説明
abs(n)	n:数値	数値	絶対値を返す
floor(n)	n:数値	数値	少数以下を切り下げた整数値を返す（ 切り上げ ceil 、四捨五入 round ）
random()		数値	0.0～1.0 までのランダムな数値を返す
pow(n,m)	n:数値 m:数値	数値	n の m 乗を返す
exp(n)	n:数値	数値	e の n 乗を返す
log(n)	n:数値	数値	e を底とした n の対数を返す
sqrt(n)	n:数値	数値	n の平方根を返す
sin(x)	x:ラジアン	数値	x の正弦を返す（-1.0～1.0）※ラジアン = 角度/180 * Math.PI
cos(x)	x:ラジアン	数値	x の余弦を返す（-1.0～1.0）
tan(x)	x:ラジアン	数値	x の正接を返す（-∞～∞）
atan2(y,x)	y:数値 x:数値	数値	座標 x,y の角度を返す

10.2 String

文字列操作関連のオブジェクト。

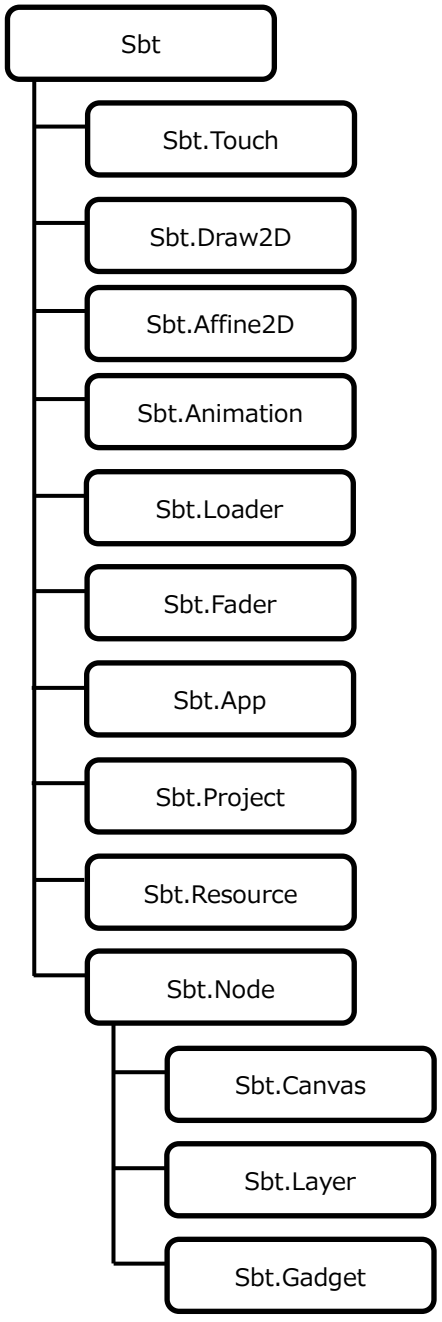
10.2.1 メソッド

メソッド内の s は、文字列オブジェクトを指します。

メソッド	引数	戻り値	説明
s.length		数値	s 文字列の長さを返す
s.substring(top [,end])	top:始点 end:終点	文字列	s 文字列内の始点から終点までの文字列を返す
s.slice(top [,end])	top:始点 end:終点	文字列	substring()と基本は同じだが、top にマイナス値を入れると文字列の終わりからの位置を指定できる
s.substr(top [,len])	top:始点 len:文字数	文字列	s 文字列内の始点から長さ分の文字列を返す（始点にマイナスを入れると文字列の後ろからの位置を指定できる）
s.split(ch [,max])	ch:区切り文字 max:最大個数	文字列配列	区切り文字ごとに文字列を切り出して文字列配列を返す
s.concat(str)	str:文字列	文字列	s 自身と与えられた文字列を結合した文字列を返す

11 Sbt クラス

プチデベロッパーで作成したアプリケーションの実行時に組み込まれるスマイルブーム製のライブラリです。



Sbt	ライブラリ本体
Sbt.Touch	タッチ・マウスイベント情報を管理するクラス
Sbt.Draw2D	描画を制御するサポートクラス
Sbt.Affine2D	2D アフィン変換に関するサポートクラス
Sbt.Animation	アニメーションを制御するサポートクラス
Sbt.Loader	ファイル読み込みに関するサポートクラス
Sbt.Fader	画面のフェード処理のサポートクラス
Sbt.App	実行エンジン
Sbt.Project	プロジェクト情報を管理するクラス
Sbt.Resource	リソース情報を管理するクラス
Sbt.Node	表示物の基底クラス
Sbt.Canvas	Canvas 表示に関するクラス
Sbt.Layer	Layer 表示に関するクラス
Sbt.Gadget	Gadget 表示に関するクラス

お問い合わせ先

本製品に関するお問い合わせは、公式ページのお問合せフォームからお願いいたします。

http://smileboom.com/special/sbpd/html_form.php

制作・著作　株式会社スマイルブーム
〒060-0809　北海道札幌市北区北９条西４丁目７番４号エルムビル
許可なく複写・転載を禁止します。